Searching for rare traffic signs

Master's thesis in Computer science and engineering

Hannes Gustafsson

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
UNIVERSITY OF GOTHENBURG
Gothenburg, Sweden 2021

Master's thesis 2021

Searching for rare traffic signs

Hannes Gustafsson

UNIVERSITY OF
GOTHENBURG

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Computer Science and Engineering
Chalmers University of Technology
University of Gothenburg
Gothenburg, Sweden 2021

Searching for rare traffic signs

Hannes Gustafsson

Supervisor: Bernhard Mehlig, Department of Physics
Advisor: Erik Werner & Niklas Gustafsson, Zenseact
Examiner: Bernhard Mehlig, Department of Physics

Master's Thesis 2021
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: The cover depicts a magnifying glass over a traffic sign.

Typeset in LaTeX
Gothenburg, Sweden 2021

Searching for rare traffic signs

Hannes Gustafsson
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg

Abstract

Deep neural networks are good at recognizing traffic signs when they are trained on many different examples of each one. However, some traffic signs are very rare and not often encountered when collecting data. This means that a network does not recognize rare traffic signs as well as those that are encountered often. When collecting large amounts of data, one usually only labels a small subset of it. Therefore, there might exist more examples of the rare traffic signs in the unlabeled data set. If these examples could be found and used in training, the performance of the model could be expected to improve. This thesis evaluates how a standard neural network performs in searching for rare traffic signs, and whether some commonly used techniques from few-shot learning can improve its performance. To our surprise we find that they cannot. Furthermore, in this thesis we show that searching for rare traffic signs is an efficient active learning method, outperforming other established methods by requiring up to 8x less additional data to achieve the same F1-score on rare traffic sign recognition.

Keywords: Computer, science, computer science, engineering, project, thesis, deep learning.

Acknowledgements

I would like to thank my supervisor Bernhard Mehlig for his support and guidance through out the thesis project. I would also like to thank my supervisors and colleagues from Zenseact Erik Werner, Niklas Gustafsson and Sami Jaghour for their continuous help in the proejct.

Hannes Gustafsson, Gothenburg, August 2021

Contents

List of Figures

1

Introduction

Neural networks have revolutionized computer vision. Deep neural networks excel at locating and classifying objects in images, and they are now the standard choice for perception systems in self-driving cars [2, 3].

However, neural networks work best when they are trained on very large data sets that contain many examples of each object to classify [16]. Usually this means that neural networks struggle to recognize rare objects that occur only a few times in the training set.

The main data set that we use in this thesis is the Mapillary traffic sign data set, which consisting of images of traffic signs from all over the world [13]. Some types of traffic signs in the data set are very rare and are therefore not often encountered when data is collected. This means that the frequency for the rare types of traffic signs in the data set is low and the frequency of common traffic signs is high. Examples of rare types of traffic signs are presented in figure 1.

While the Mapillary traffic sign data set has class labels associated with each traffic sign, it is commonly the case that there are many more unlabeled images than one can afford to obtain labels for. If we randomly select a subset of images for which to acquire labels, images that are rare in the unlabeled data set will also be rare in the labeled subset. However, if we select images depicting rare classes for labeling, we can make the rare classes more common in our labeled data set. As a result, a neural network trained on this labeled data set can be expected to be better at recognizing rare classes than a network trained on the randomly selected subset.

The question is, how can we identify the images containing the rare classes, so that we can label them? One could let a human search for them in the data set of unlabeled images. This is however unfeasible, due to the large size of the unlabeled data set. A method that can decrease the number of images that a person has to sort through should be developed.

Figure 1.1: Examples of rare traffic signs from Mapillary traffic sign data set.

The purpose of this thesis project is to develop such an automatic method that can find images of rare types of traffic signs in a large unlabeled data set. Further, we test if labeling and adding the found images to the training set is an efficient way of improving a model's final performance on the rare traffic signs.

1.1 Aim

In this thesis project we aim to develop an algorithm that can find specified types of rare traffic signs in an unlabeled data set. Our search algorithm returns a set of images that a neural network predicts to be most likely to depict the specific type of traffic sign. The algorithm will be trained using a training set that contains few labeled images of rare traffic signs but many labeled images of common traffic signs.

One straightforward way to do this would be to train a standard neural network to classify the rare traffic signs using the few labeled images available in the training set. However, standard neural networks need lots of training data to detect objects well, which is not available in this case since the traffic signs are rare. We hypothesize that we can improve a model's ability of searching for rare traffic signs by using techniques from the field of *few-shot learning* [9].

After we have found images of rare traffic signs in the unlabeled data set we label them and move them to the training set. We hope that this is an efficient way to improve a model's accuracy on the rare types of traffic signs. The concept of letting a learning algorithm influence the data it's trained on is called *active learning* [10]. The second goal of this thesis is to compare the method of searching for rare traffic signs to other active learning methods.

1.1.1 Few-shot learning

Few-shot learning methods aim to improve classification accuracy on types of objects where one only has few labeled examples [9]. A common way to to this is to learn an embedding such that images that depict the same type of object have a a similar embedding, while images that depict different objects have a dissimilar embedding. This is called metric-based few-shot learning [9]. In this thesis we test two metric-based few-shot learning methods, ProtoNet and RelationNet. These methods are explained in section 3.2.

1.1.2 Active learning

Active learning methods are algorithms that let neural networks decide which images should be labeled and used for training [10]. Active learning is often performed iteratively where one has a labeled data set and a unlabeled data set. In each iteration a part of the unlabeled data set is selected by an active learning method, annotated, and moved to the training set. The difference between active learning methods is how they select which samples to label. A common active learning method is to select the samples in the unlabeled data set where the model shows the

highest degree of uncertainty. [10]. To compare different active learning methods one measures how performance increases on a fixed validation set as the different methods select new samples that a model is trained on.

1.2 Approach and research questions

This thesis describes an algorithm that is used to search for rare types of traffic signs in a large unlabeled data set. Although the model will not be allowed to see the ground truth labels for the images in the unlabeled data set, we have access to them. This means we can easily measure the algorithm's search efficiency. We measure search efficiency using a metric called S-score, defined in section 3.3.

The baseline method we use for searching for rare classes is to train a neural network to classify the images in the labeled set, applying this network to the images in the unlabeled set, and select the images that the images classifies as most likely to belong to the rare classes. Since we have so few examples of the rare classes in the labeled set, we expect that using few-shot learning techniques can lead to a model that has better performance on the rare classes, which we in turn expect to yield a higher search efficiency.

Finally, we measure how well a search-based active learning method compares to traditional active learning methods such as uncertainty based ones. We measure this by comparing how the F1-score improves by using the different active learning methods over iterations.

In summary, the research questions that we wanted to answer in this project are:

1. Does replacing a standard neural network model with a few-shot learning model improve search efficiency?
2. Does using search as an active learning method improve precision and recall more compared to traditional active learning methods.

1.3 Limitations

Due to time constraints we limit ourselves to two distinct few-shot learning algorithms, ProtoNet and RelationNet [11, 12]. Further, we measure our performance on two datasets, the Mapillary traffic sign data set, and MiniImageNet [13, 14].

2
Theory

This section of the thesis aims to give the reader a theoretical introduction to neural networks and associated techniques. We begin by giving a brief introduction to neural networks. We then describe metrics that are used in this thesis project.

2.1 Artificial neural networks

Artificial neural networks are computing systems that are inspired by biological neural networks [18]. The network consists of neurons and weights that connect them. All though there exists many different designs of neural networks, in this thesis two of the most common ones will be used, fully-connected and convolutional neural networks.

2.1.1 Fully-connected networks

Fully-connected feed-forward neural networks are described in [4, 18]. A fully-connected feed-forward neural network can be seen as directed, acyclic graph [4]. The graph is made up of layers, containing nodes, where each node in layer l has an edge connecting it to each node in layer $l+1$. Each edge has a value called a weight that is trainable. We denote the weight between node i in layer l and node j in layer $l-1$ as w_{ij}^l. Each node has a value that is computed as

$$z_i^l = \sum_j v_j^{l-1} w_{ij}^l + b_i^l$$

where v_j^{l-1} is the output of neuron j in layer $l-1$ [18]. v_i^l is computed by applying an activation function, g, on z_i^l. An activation function, g, is a continuous non-linear function. This calculation is performed for each neuron in the layer. After the output has been calculated for a layer, one continues by calculating the output of the subsequent layer. The final layer of the network is the output of the network. Figure 2.1 illustrates the neural network as a directed graph.

2.1.2 Convolutional neural networks

Convolutional neural networks are described in [18, 4]. The following section is a brief summary. Convolutional neural networks (CNN) are a specialized type of neural networks that are often used to process input data with a natural grid structure, such as images. The network was introduced in [6].

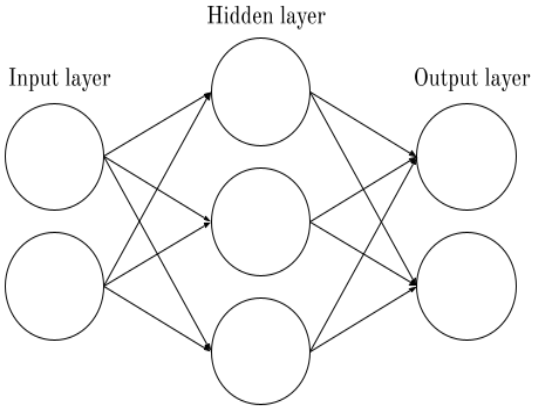


Figure 2.1: Illustration of a feed-forward neural network.

Convolutional neural networks have shared weights between neurons in each layer and often a local receptive field. This is beneficial in two main ways, combating overfitting, due to fewer total number of weights and effectively expanding the training set. One can think of a CNN as a filter that is applied to an input grid in a sliding window approach. The filter usually has a local receptive field such that it only receives inputs from a small part of the full input grid.

Each CNN filter is described by a set of hyper parameters, input depth, width and height (D, W, H). A single convolutional neural network with these hyper parameters will contain $D \times W \times H$ of trainable parameters plus a bias parameter. The output of a CNN filter applied to an input with a single dimension is calculated as following:

$$y_{ij} = \sum_{k=1,l=1}^{K,l} w_{kl} x_{k+(i-1),l+(j-1)}$$

Here, w_{kl} is the trainable parameter at position k, l of the CNN. As in a feed-forward neural network one also applies an activation function on the output of the CNN. These sorts of network have been shown to work well on image tasks such as object detection and image classification. The output depth of the neural network will be equal to the number of filters applied to the input.

2.1.3 Supervised learning

In supervised learning, a model is trained to give output that is close as possible to a predefined ground truth [4]. One of the most common supervised learning tasks is classification.

The task of classification is to group inputs to a set of predefined categories, or classes [4]. The inputs can for instance be images or text sequences. Each image or text sequence belongs to a certain class and each output node in the network

represents a single class. The ground truth for classification task is a vector that has the same number of elements as there are numbers of predefined classes. Only a single element in the vector has a value of 1 and the rest are equal to 0.

2.1.4 Transfer learning

Transfer learning is a machine learning method where a model that has been trained on a data set X is used as a starting point to train on data set Y [17]. The aim is to transfer the knowledge that it has learned from X to Y. Experiments have shown that models that have been trained on large image data sets before being trained on smaller image data sets can achieve high performance faster compared to only training on the small data set [17].

2.2 Training a neural network

Training a neural network means adjusting the values of the trainable parameters. This is often done by gradient based optimization algorithms such as stochastic gradient descent [4]. Gradient descent updates each weight of a network as follows:

$$w_{ij} = w_{ij} - \gamma \frac{\partial L}{\partial w_{ij}}$$

Where L is a loss function and γ is a hyper parameter that specifies the step size for each weight update. To calculate $\frac{\partial L}{\partial w_{ij}}$ one uses back propagation.

2.2.1 Loss functions

The loss function is what the model is trained to minimize [4, 18]. The loss function is associated with the goal of the task, such that lowering the loss increases the performance of the network on the given task. The loss is a function that depends on the model, an input and a target output such that it is differentiable with regards to the model parameters.

A common loss function for classification is cross entropy which is defined as

$$L = -\sum_{i}^{I} y_i \log f_i(x)$$

Here x is the input, $f_i(x)$ is the output of neuron with index i of the model, and y_i is the ground truth value, for the element with index i in the one-hot encoded vector, corresponding to the input x. For one-hot targets, thus further simplifies to

$$L = -\log f_k(x), \tag{2.1}$$

where k is the index corresponding to the ground truth class.

2.2.2 Back-propagation

Back-propagation was introduced in [7]. Back-propagation is a technique to calculate the gradients of the loss function with regards to the model's trainable parameters [7]. Back-propagation computes the gradient using the chain rule, computing one layer at a time.

2.3 F1-score, precision and recall

In this thesis precision, recall and F1-score are used as metrics [5]. Precision is calculated as:

$$\text{Precision} = \frac{TP}{TP + FP}$$

and recall is calculated as:

$$\text{Recall} = \frac{TP}{TP + FN}$$

Here, TP is number of true positive, FP is number of false positive and FN is number of false negative. True positive is when a model predicts that an input belongs to a class and the ground truth agrees. False positive is when a model predicts that an input belongs to a certain class and the ground truth does not agree. False negative is when a model predicts that an input does not belong to a certain class while the ground truth says that it does belong to that class.

F1-score is calculated by combining precision and recall as:

$$\text{F1-score} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

3
Methods

In this section the methods used in the project are presented.

3.1 Data

In this section we present the data that is used to perform the experiments.

3.1.1 Mapillary Traffic Signs

Mapillary's traffic sign data set is used in two experiments [13]. The data set contains images of 313 different types of traffic signs from all over the world. Some traffic signs are very rare in the data set, such as those shown in figure 1.1, which makes it suitable for the purpose of this thesis. The full frequency distribution of the data set is shown in figure 3.1.

Some images of traffic signs are very small and might introduce noisy signals into the network. These small images are hard to recognize, even for humans. Therefore, all images that have an area smaller than 16^2 pixels are removed from the data set. After filtering, the full data set contains 59,195 images of 313 types of traffic signs. 18,564 images are used for training the models, 2,565 images are used to validate them and finally 38,066 images are used to act as the large unlabeled data set, where we search for rare traffic signs.

3.1.2 MiniImageNet

We also perform one experiment using the MiniImageNet data set in order to test how well the methods transfer to very different data sets [14]. MiniImageNet is a subset of the ImageNet data set [20]. It contains 93 different classes such as dogs, cats and humans. Each class has 600 images depicting the class. As with the traffic sign data set we split the full data set into train, validation and test set, where the test set acts as the unlabeled data set, where we search for rare classes. The train set contains 14,400 images, the validation set 1,600 images and the test set 37,000 images. MiniImageNet only contains relatively large images and therefore no filtering due to small sizes is needed.

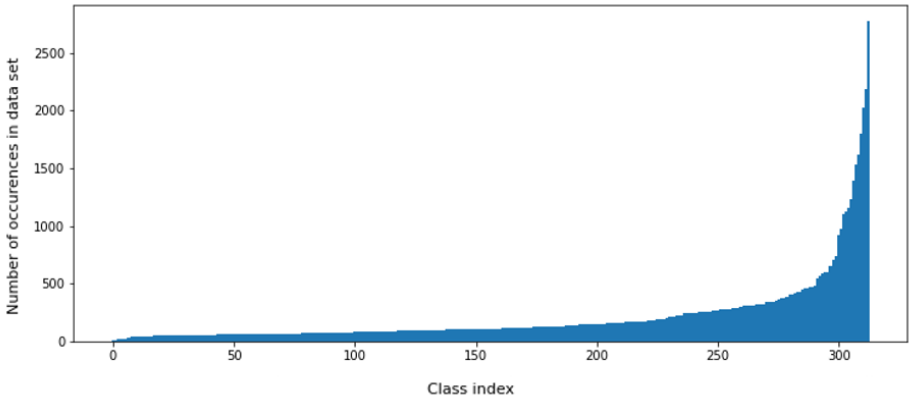


Figure 3.1: The y-axis shows how many images are present in Mapillary traffic sign data set for a given type of traffic sign. The x-axis is the category index for that traffic sign. The left most types of traffic signs are the very rare types of traffic signs that the models are queried to search for.

3.2 Models

In this thesis three different neural network models are tested and compared. All models are implemented in the PyTorch framework [8]. This section aims to explain each model. For the experiments that do not involve the MiniImageNet data set all models are pretrained on the ImageNet data set in order to speed up training. All models share a common convolutional neural network structure called ResNet-18 [15]. This architecture was chosen due to the high accuracy it achieves on ImageNet while also being quite fast to train.

The models that we use have been pretrained to predict which of 1,000 classes an image depicts. Since we are not interested in these predictions we remove the output layer. By removing the original output layer the original model's second to last layer will become the output layer. If we now feed an image to the model it will not output class probabilities, instead it will produce a 512 dimensional vector that represents the content of the image. We call this vector a feature vector.

3.2.1 ProtoNet

ProtoNet is based on a neural network f, which in this thesis is a ResNet-18 [11]. By passing an image x_i into f a feature vector is produced. Before each training batch one calculates a mean feature vector for each class c_k. This is done using a randomly sampled subset, C_k, of images labeled as c_k:

$$P_{c_k} = \frac{1}{|C_k|} \sum_{x_i \in C_k} f(x_i)$$

f is trained to minimize the distance between $f(x_i)$ and P_{c_k} for all x_i that are labeled as the class c_k. Simultaneously, f is also trained to maximize the distance between $f(x_i)$ and the mean feature vectors for all other classes. This training objective is illustrated in figure 3.2.

The loss function that ProtoNet is trained to minimize is

$$L(x) = d(f(x), P_i) + \log \sum_{j \neq i} \exp(-d(f(x), P_j)).$$

Here P_j is the mean feature vectors for classes that x does not belong to and P_i is the mean feature vector for the class that x does belong to. $d(x, y)$ is the Euclidean distance between x and y.

During test time one calculates the mean feature vector for a class r_k using a set, R_k, of images labeled as r_k:

$$P_{r_k} = \frac{1}{|R_k|} \sum_{x_i \in R_k} f(x_i)$$

The model classifies an unlabeled image x_u based on its feature vector's, $f(x_u)$, distance from P_{r_k}.

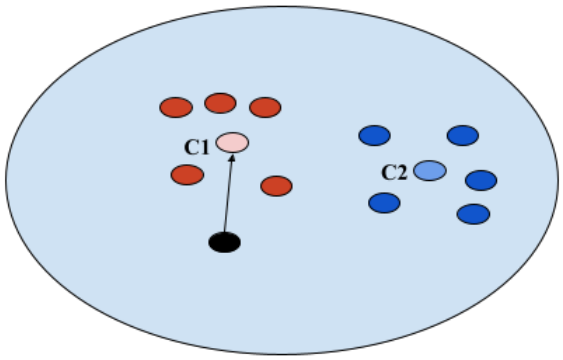


Figure 3.2: Illustration of the task that ProtoNet is trained to solve. The blue background shows the feature space. The dark red circles show the projected feature vectors of five images, all of which belong to the class C1. The bright red circle shows the average of the dark red circles. The model is trained to produce a feature vector for all images belonging to C1 to be close to the bright red circle and far away from the light blue circle. The black circle is the projected feature vector of an unlabeled traffic sign. In this case it is classified as belonging to C1 since the center of C1 is closer than C2.

3.2.2 RelationNet

RelationNet is based on two neural networks, a feature extractor f and a relation module g [12]. In this thesis f is the ResNet-18. The relation module g has a single output neuron. During training each image containing a traffic sign is fed into f such that a feature vector is produced. Two different feature vectors are concatenated $[f(x_i) : f(x_j)]$ where x_i and x_j can either depict the same type of traffic sign or two different. The concatenation, $[f(x_i) : f(x_j)]$, is then passed through the regression module, g. The relation module is trained to have a large output value if x_i and x_j depict the same type of traffic sign, and low otherwise. The model is trained to minimize the mean squared error:

$$L(x_1, x_2) = (g([f(x_1) : f(x_2)]) - y)^2$$

where y is one if x_1 and x_2 depict the same object, and zero otherwise. In figure 3.3 an illustration of RelationNet's architecture is presented.

Note that the model never predicts which class an image belongs to, only whether two images belong to same class. Therefore, during test time one of the two images, passed through the model, will be labeled while the other is not. The model will assign the unlabeled image the same label as the labeled image that it produces the highest relation score with.

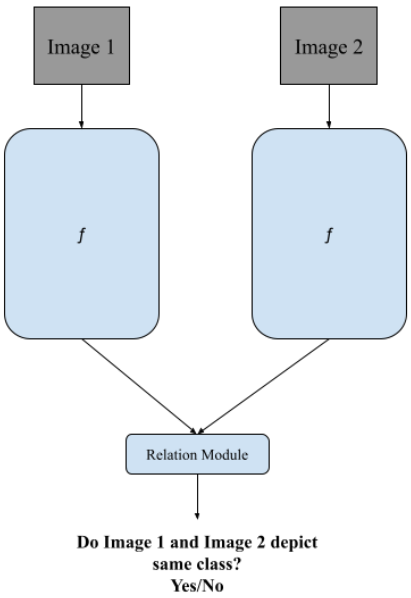


Figure 3.3: Illustration of the task that RelationNet is trained to solve.

3.2.3 StandardNet

StandardNet is used as the baseline. The model is a standard ResNet-18 with a final output layer consisting of 313 nodes. Each output node represents a single type of traffic sign. The model is trained such that given an image of a traffic sign of category C, the output node corresponding to category C should have the highest value of all output nodes. For StandardNet a softmax activation function is applied to the final layer [4]. The model is trained using categorical cross entropy thus minimizing:

$$L = \sum_i -y_i \log f_i(x)$$

where $f_i(x)$ is the value of the output node with index i. y_i is one if x depicts a class whose output node is i and zero otherwise.

3.2.4 Probability assignment

In the experiments, each image in the large unlabeled data set will be assigned a value corresponding to how likely it is that it depicts the same traffic sign as the one searched for. The method of assigning the values is different depending on which model is used.

For StandardNet each image in the unlabeled data set will be assigned the value of the output neuron that corresponds to the queried traffic sign. The value corresponds to the networks estimated probability that the image belongs to the queried traffic sign.

For ProtoNet each image is assigned the distance between the feature vector of the queried traffic sign and the feature vector of the sample. A smaller distance indicates higher probability of depicting same type of traffic sign as the queried.

For RelationNet the value assigned to each image is the relation score between the queried traffic sign and the sample.

3.3 S-score

In the first experiment the S-scores are tracked for all models and setups. The S-score measures how well the model performed during search for the queried sample. We define the S-score as

$$\text{S-K} = \frac{M}{\min(K, N)}$$

where M is the number of correct samples in top-K, K is an integer and N is total number of correct samples in the unlabeled data set. Thus, if the model found 3 correct images in the top 50 and there existed 30 total correct images in the unlabeled data set then the S-50 score would be $\frac{3}{\min(50,30)} = \frac{3}{30} = 0.1$

3.4 Few-shot learning experiment

In this experiment we test the hypothesis that replacing a standard neural network with a few-shot learning model will improve search efficiency. Three different models are tested: StandardNet, ProtoNet and RelationNet.

At the beginning of the experiment 25 types of traffic signs are chosen as the rare traffic signs that the models will search for. A single sample of each type of rare traffic sign can be seen in figure 3.4. For each of the 25 types of rare traffic signs there exists between 5 and 81 images in the unlabeled data set that depict the same rare traffic sign. Since the number of images in the unlabeled data set is 38 066 the probability of randomly choosing a rare traffic sign is small.

Figure 3.4: A single sample of each type of rare traffic sign used in the few-shot learning experiment.

All images of the chosen traffic signs, except for $N = 1, 5, 10$, depending on the experiment setup, are moved from the training set to the large unlabeled data set. Therefore all three models will only have a small set of examples of the rare traffic signs available during training. This is done to imitate the situation where one only has access to a few samples of the rare traffic sign.

Before searching, each model is trained on the remaining training set for 40 epochs and evaluated on the validation set.

To search for the rare traffic signs, the models are used to assign a value to each sample in the large unlabeled data set. The value assigned to each sample indicates how likely it is that it depicts one of the rare traffic signs. All samples in the large unlabeled data set are then sorted according to their assigned values. Finally, the top-K samples in the sorted list of samples are the model's proposals. For the experiment we measure the average S-score over the set of rare traffic signs with $K = 5, 20, 100, 1000$.

3.4.1 Few-shot learning experiment on MiniImageNet

Apart from the traffic sign data set we also perform the same experiment on the MiniImageNet data set in order to see how well the methods transfer to a different data set. The same setup is used as above. MiniImageNet is well balanced and therefore the classes that are treated as rare objects are selected randomly.

3.5 Active learning experiment

In the active learning experiment we test how searching for rare traffic signs compares to other active learning methods.

The experiment is setup as follows. A set of rare traffic signs is chosen and all images that depict them, except for one from each class, are removed from the training set. In this experiment 6 different types of traffic signs are used.

We use two different setups with regards to how rare the 6 types of traffic sign are in the unlabeled data set. In the first setup we make no changes to the unlabeled data set. Thus, for each of the six rare traffic signs there exist between 150 and 250 images in the unlabeled data set that depict the same sign. For the second setup we limit the number of samples for each of the 6 types of traffic sign to a maximum of 50 samples per type. This is done in order to see how the active learning methods performance changes with how rare the 6 traffic signs are.

The experiment is performed in iterations. Each iteration consists of two steps, training and querying. A total of 10 iterations are performed per active learning method. At the start of each iteration a StandardNet model is trained using the current labeled data set and is evaluated on a static validation set of traffic signs. Note that at the first iteration only a single sample per rare traffic sign is present in the training data set.

After training the model, a set of images from the unlabeled data set is chosen according to one of five active learning metrics. The chosen images are labeled and added to the training set. Pseudo code for the full process is presented in listing 1.

We track how a StandardNet's F1-score on the rare traffic signs improves as more data is added the the training set.

3.5.1 Active-learning methods

In this experiment five different active learning methods are compared.

- Choosing the most likely samples to belong to a rare traffic sign according to (1) ProtoNet, (2) RelationNet and (3) StandardNet.
- (4) Choosing the samples that have the highest uncertainty according to output probability .
- (5) Choosing samples at random.

The three search models are trained using same setup as in the earlier experiment. When choosing the samples that should be labeled and used for training in the next iteration, the model's top 50 proposals for each of the 6 types of traffic signs are used. In total this amounts to 300 images being added per iteration. The search models are also retrained using the new train set.

The uncertainty method chooses the 300 samples in the unlabeled data set where the model displays the highest level of uncertainty. Uncertainty is measured according to the following formula.

$$\text{Entropy}(x) = \sum_{i=1}^{I} -f_i(x) \log f_i(x).$$

Here $f_i(x)$ is the probability that x depicts the traffic sign corresponding to the output node with index i, according to a StandardNet model, and I is the total number of classes. All samples in the unlabeled data set are sorted according to this method.

For the random method, 300 random images from the unlabeled data will be chosen and added to the training set.

4
Results

In this section results for the experiments introduced in section 3 are presented.

4.1 Few-shot learning experiment

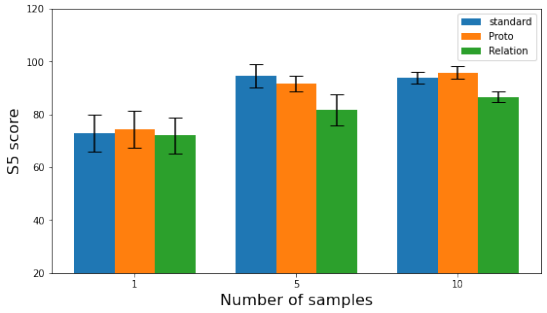
In this section we present the results that are aimed to answer the first of our research questions. Does replacing a standard neural network model with a few-shot learning model improve efficiency? The few-shot learning experiment was carried out using two data sets, Mapillarys traffic sign data set and MiniImageNet. We first present the average S-scores for searching for rare traffic signs, followed by searching for rare objects from the MiniImageNet data set.

4.1.1 Traffic sign data set

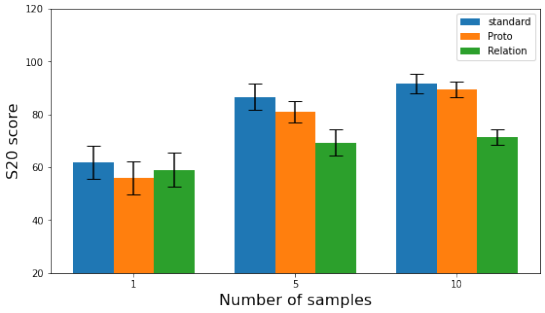
In this section the results for the few-shot learning experiment on the traffic sign data set are presented.

4.1.1.1 Quantitative

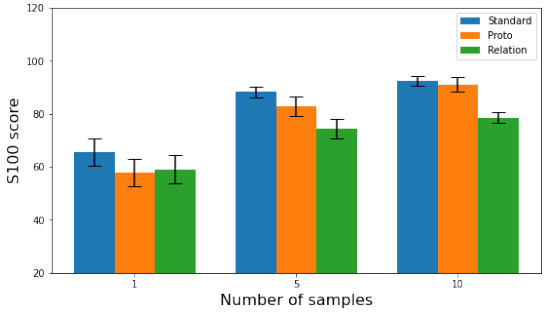
Each graph shows the average S-score on the y-axis and the number of labeled images per type of traffic sign available during training on the x-axis. The experiment was done using 25 types of traffic signs. The standard errors or the mean are also presented for each model and initial sample size.



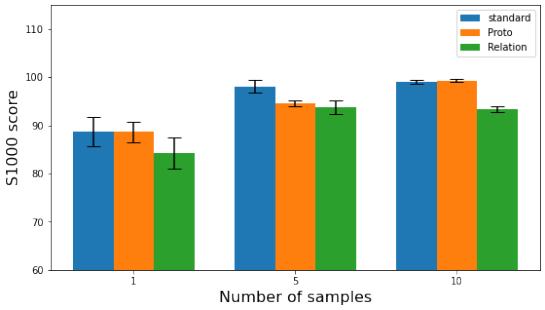
(a) Average S5-Score



(b) Average S20-Score



(c) Average S100-Score



(d) Average S1000-Score

Figure 4.1: The colored bars show average S-scores on the traffic sign data set for three different sample sizes, 1, 5 and 10. Sample sizes are the number of images of the rare traffic signs that are available during training. The three colors represent the three models. The vertical black lines on top of all bars show the standard error of the mean.

4.1.1.2 Qualitative

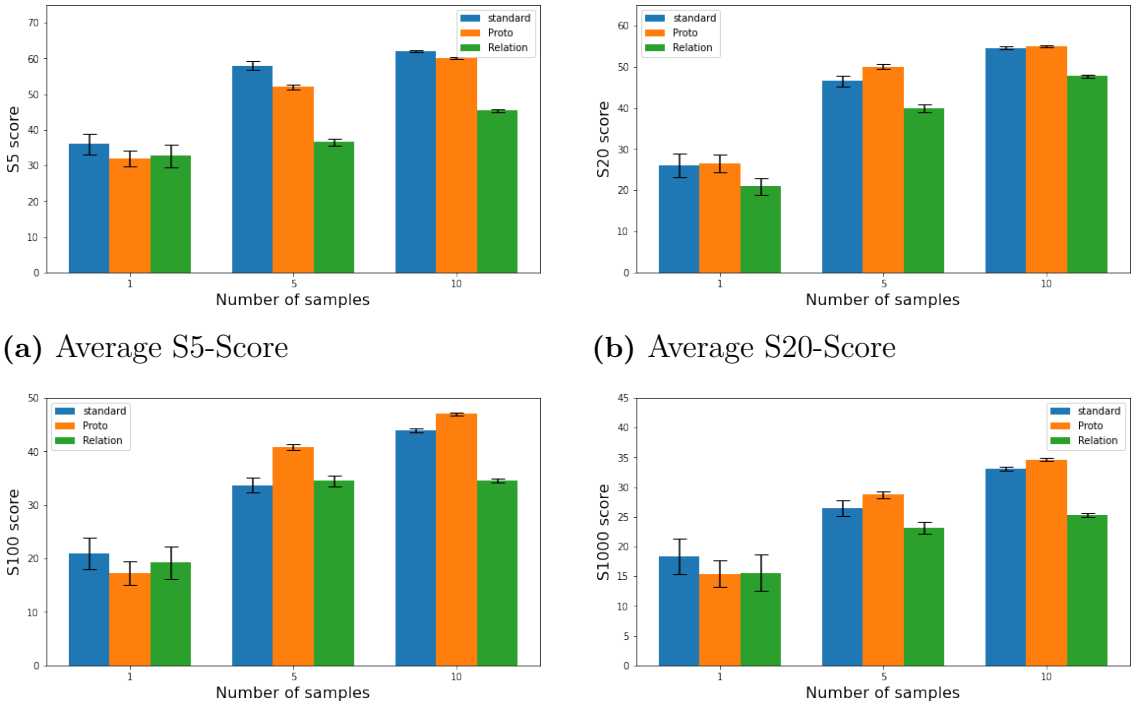
In this section we present qualitative results of searching for two different rare traffic signs using StandardNet. The results show the top-20 proposals given by StandardNet while searching for two types of rare traffic signs. In both examples StandardNet is only trained using a single example of the two rare traffic signs.

(a) **(b)**

Figure 4.2: The two figures above show the top-20 proposals given by StandardNet. Above each of the 40 images are the assigned probability to each given sample. At the bottom left of the two figures the single training sample that was available to the model are presented.

4.1.2 MiniImageNet

In this section the results for the first experiment on the MiniImageNet data set are presented. Each graph shows the average S-score on the y-axis and number of labeled images that was available during training per class on the x-axis. The experiment was done using 10 unique classes. The score was calculated by averaging the given S-score over all classes per model and sample size. The standard errors are also presented for each model and initial sample size.



(a) Average S5-Score

(b) Average S20-Score

(c) Average S100-Score

(d) Average S1000-Score

Figure 4.3: The colored bars shows the average S-scores on the MiniImagenet data set for three different sample size, 1, 5 and 10. Sample size is number of images of the searched for traffic sign that are available during training. The vertical black lines on top of all bars shows the standard errors of the mean. The three colors represents the three models.

4.2 Active learning experiment

In section 3.5 we stated that we wanted to evaluate search as an active learning method. In this section, we present the results of an experiment where we aim to answer if search is a more efficient active learning method than other established ones. Each line in the two figures below represents a single active learning method. The search based active learning metrics are labeled as Proto, Standard and RelationNet and the established active learning methods are labeled as Entropy and Random. In figure 4.4 the F1-scores on the chosen rare classes for StandardNet and how they improve as more data is added to the training set is presented. In figure 4.5 we show the F1-scores of the chosen rare classes when we have limited the number of samples of the 6 types of traffic signs, making them even more rare. For each iteration 300 new data points are added to the training set.

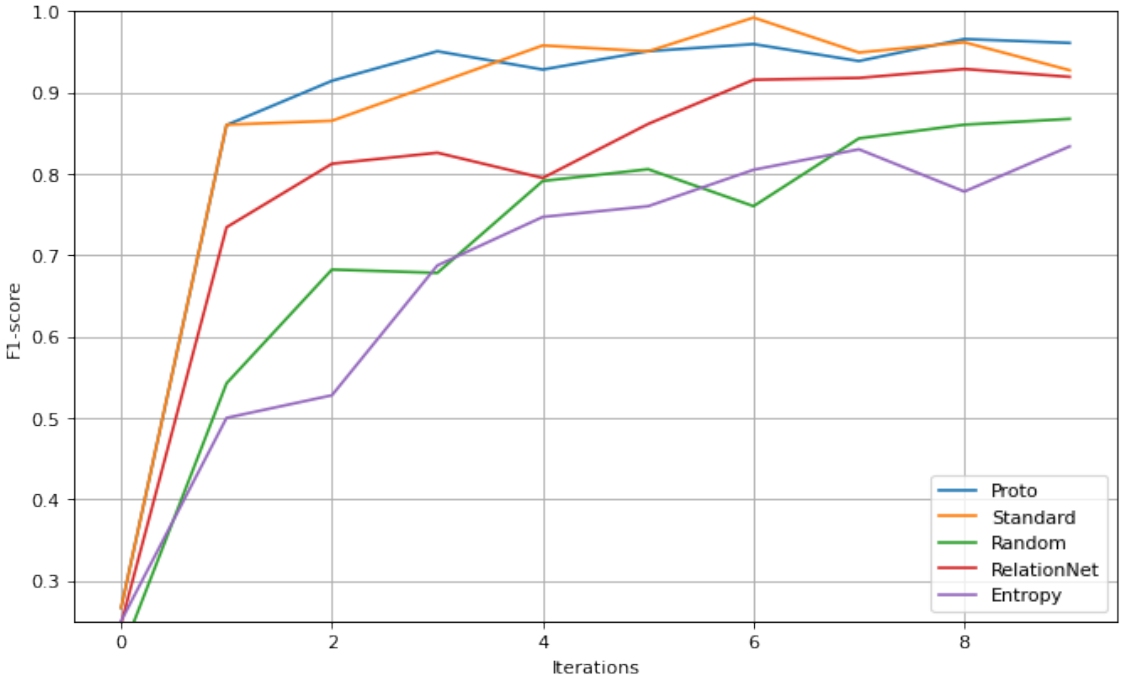


Figure 4.4: The figure above shows the results for the active learning experiments with uncapped rarity. The x-axis is number of active learning iterations that has been performed. The y-axis shows the F1-score of a StandardNet trained on the samples chosen by each metric at each iteration.

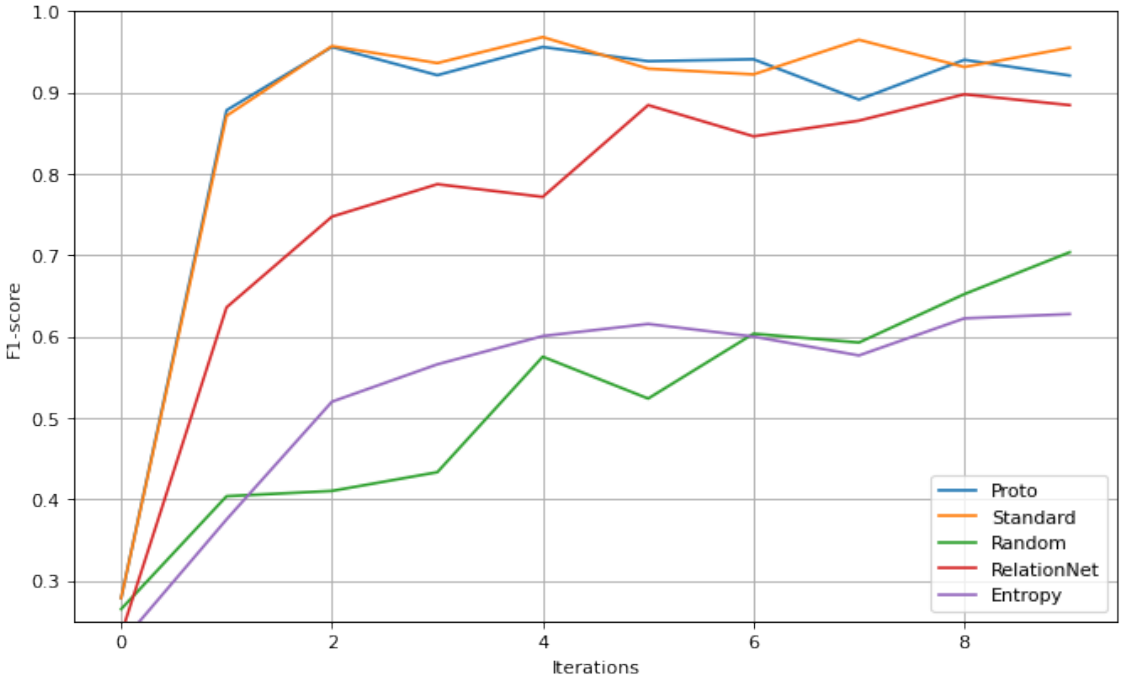


Figure 4.5: The figure above shows the results for the active learning experiments with capped rarity. The x-axis is number of active learning iterations that has been performed. The y-axis shows the F1-score of a StandardNet trained on the samples chosen by each metric at each iteration.

5
Conclusion

The aim of this thesis project was to develop a method that could find images of rare types of traffic signs in a large unlabeled data set. We also aimed to investigate if labeling and adding the found images of rare types of traffic signs was an efficient way of improving a models accuracy on rare types of traffic signs. We therefore formulated the following two research questions that we aimed to answer.

1. Does replacing a standard neural network model with a few-shot learning model improve search efficiency?
2. Does using search as an active learning method improve precision and recall more compared to traditional active learning methods?

To answer the first question we implemented two few-shot learning models, ProtoNet and RelationNet, as well as one standard neural network. All three were used to search for rare traffic signs in a large unlabeled data set. Our results showed that using few-shot learning models had only a small impact on search efficiency, and that a standard neural network is in the majority of cases equally efficient or even more efficient.

The models were also used to search for other, more abstract and difficult to recognize objects, in an unlabeled data set of images from MiniImagenet. Our results showed that search efficiencies for all models were heavily affected and decreased by more than 50 %. The results showed that searching for more difficult to recognize objects using few-shot learning models did not have a big impact on the S-score as StandardNet achieved comparable efficiency as both ProtoNet and RelationNet.

To answer the second question we compared search based active-learning methods with other established methods. This was done by iterativly adding data to a training set using the methods and evaluating the effects the new data points had on a model. Our results showed that using a search based active learning method is a very efficient way to improve performance on traffic signs that are very rare in the training set. The search methods outperformed standard active learning methods by needing up to 8 times fewer additional data points to achieve similar F1-score. This can be seen in figure 4.4 and 4.5 where the search based active learning methods need only a few iterations worth of additional data points to achieve an F1-score of around 0.9.

Our results also showed that the rarer the traffic signs were, the better the search

methods performed compared to the other active learning methods. Comparing figure 4.4 and figure 4.5, we can see that the Random and Entropy methods perform much worse when one makes the rare traffic signs even more rare. However, this does not occur for the search based active learning methods, which perform almost equally well on both setups.

Finally we will discuss questions for future work. Our tests on the MiniImageNet and the Mapillary traffic-sign data sets show that the search efficiency can vary substantially between two data sets. It would be interesting to conclude how well the search methods presented in this thesis could transfer to even larger and more unbalanced data sets.

The models used for experiments, involving Mapillary traffic-sign data, were pretrained on the standard ImageNet data set. It would be interesting to see how well search efficiency would scale with models that were pretrained on data sets even larger than ImageNet.

Bibliography

[1] Frisk, D. (2016) A Chalmers University of Technology Master's thesis template for LaTeX. Unpublished.

[2] Niall O' Mahony, Sean Campbell, Anderson Carvalho, Suman Harapanahalli, Gustavo Velasco Hernandez, Lenka Krpalkova, Daniel Riordan, Joseph Walsh. Deep Learning vs. Traditional Computer Vision. *arXiv preprint arXiv:1910.13796*, 2019.

[3] Ekim Yurtsever, Jacob Lambert, Alexander Carballo, Kazuya Takeda. A Survey of Autonomous Driving: Common Practices and Emerging Technologies. *arXiv preprint arXiv:1906.05113*, 2019.

[4] Goodfellow, I., Bengio, Y. and Courville, A. Deep Learning. pp.98,168,182,327-329. 2016.

[5] David M W Powers. Evaluation: From Precision, Recall and F-Factor to ROC, Informedness, Markedness & Correlation. 2007

[6] Kunihiko Fukushima. Neocognitron: A Self-organizing Neural Network Model for a Mechanism of Pattern Recognition Unaffected by Shift in Position. 1980.

[7] Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986) Learning representations by back-propagating errors.

[8] Paszke, Adam and Gross. PyTorch: An Imperative Style, High-Performance Deep Learning Library. *arXiv preprint arXiv:1912.01703*, 2019.

[9] Yaqing Wang, Quanming Yao, James Kwok, Lionel M. Ni. Generalizing from a Few Examples: A Survey on Few-Shot Learning. *arXiv preprint arXiv:1904.05046*, 2019.

[10] Pengzhen Ren, Yun Xiao, Xiaojun Chang, Po-Yao Huang, Zhihui Li, Xiaojiang Chen, Xin Wang. A Survey of Deep Active Learning. *arXiv preprint arXiv:2009.00236*, 2020.

[11] Jake Snell, Kevin Swersky, Richard S. Zemel. Prototypical Networks for Few-shot Learning. *arXiv preprint arXiv:1703.05175*, 2016.

[12] Flood Sung, Yongxin Yang, Li Zhang, Tao Xiang, Philip H.S. Torr, Timothy M. Hospedales. Learning to Compare: Relation Network for Few-Shot Learning. *arXiv preprint arXiv:1711.06025*, 2018.

[13] Christian Ertler, Jerneja Mislej , Tobias Ollmann, Lorenzo Porzi, Gerhard Neuhold, Yubin Kuang. The Mapillary Traffic Sign Dataset for Detection and Classification on a Global Scale. *arXiv preprint arXiv:1909.04422*, 2020

[14] Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Koray Kavukcuoglu, Daan Wierstra. Matching Networks for One Shot Learning. *arXiv preprint arXiv:1606.04080*, 2016.

[15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun. Deep Residual Learning for Image Recognition. *arXiv preprint arXiv:1512.03385*, 2015.

[16] Abhinav Gupta. Revisiting the Unreasonable Effectiveness of Data. 2017.

[17] Fuzhen Zhuang, Zhiyuan Qi, Keyu Duan, Dongbo Xi, Yongchun Zhu, Hengshu Zhu, Hui Xiong, Qing He. A Comprehensive Survey on Transfer Learning. *arXiv preprint arXiv:1911.02685*, 2020.

[18] Bernhard Mehlig. Machine learning with neural networks. *arXiv preprint arXiv:1901.05639*, 2021.

[19] Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai, Joan Puigcerver, Jessica Yung, Sylvain Gelly, Neil Houlsby. Big Transfer (BiT): General Visual Representation Learning. *arXiv preprint arXiv:1912.11370*, 2019.

[20] Deng, Jia and Dong, Wei and Socher, Richard and Li, Li-Jia and Li, Kai and Fei-Fei, Li. Imagenet: A large-scale hierarchical image database. *arXiv preprint arXiv:1409.0575*, 2014.

A

Appendix

```
model = StandardNet()
AL_method = chosen_method # An active learning metric is chosen

for iteration in range(10):
        model.train(train_dataset) # Retrain
        search_model.train(train_dataset) # If applicable
        model.evaluate(validation_dataset)

        datapoints = model.choose_datapoints(unlabeled_dataset, AL_method)

        #Label the chosen data points and add to train set
        train_dataset += label(datapoints)

        #Remove chosen data points from unlabeled data set
        unlabeled_dataset -= datapoints
```

Listing 1: The pseudo code above shows how the active learning experiment is performed. The only variable that is changed during the experiment is the active learning metric. The method takes on the values Random, Entropy, StandardNetSearch, ProtoNetSearch and RelationNetSearch